#! python3

# GUI for chat client

from tkinter import \*

import socket, sys, threading,time,struct, tkinter.colorchooser, os, getpass

from tkinter import font,filedialog, messagebox

from PIL import Image, ImageTk

import tkinter.ttk as ttk

class ChatGui(Frame):

"""A frame for the flow of chat"""

def \_\_init\_\_(self, master,host, port, name):

Frame.\_\_init\_\_(self)

self.data = [StringVar(),StringVar(),StringVar()]

self.data[0].set(host)

self.data[1].set(port)

self.data[2].set(name)

self.master.title('Chat')

#self.master.resizable(0,0)

#self.master.protocol("WM\_DELETE\_WINDOW", self.\_delete\_window)

#self.master.attributes('-fullscreen', True)

self.initdir='C:/Users/'+getpass.getuser()+'/Desktop'

self.foreground="#000000"

self.background="#FFFFFF"

self.selectbackground='blue'

self.selectforeground='white'

maxchar=50

x=11

rows=10

if self.master.winfo\_screenheight()<1000:

print(self.master.winfo\_screenheight())

x=8

maxchar=35

rows=9

self.font=font.Font(family='Comic Sans MS', size=x, weight='bold', slant='roman')

self.textbox =Text(self, width =maxchar, height =rows, foreground=self.foreground, background=self.background,

selectforeground=self.selectforeground, selectbackground=self.selectbackground,font=self.font, wrap=WORD)

self.textbox.grid(row=0,rowspan=1, column=0, sticky=E)

self.scroll =ttk.Scrollbar(self, command =self.textbox.yview)

self.textbox.configure(yscrollcommand =self.scroll.set)

self.scroll.grid(column=1, row=0, rowspan=1, sticky=N+S+W)

#self.textbox=textbox(self, width=70, height=15, foreground=self.foreground, background=self.background, activestyle=NONE,

# selectforeground=self.selectforeground, selectbackground=self.selectbackground, selectmode=EXTENDED)

self.textbox.bind("<Button-3>", self.popup)

self.textbox.config(state=NORMAL)

self.textbox.insert(END, 'Хорошего дня!')

self.textbox.yview\_moveto(1.0)

self.textbox.config(state=DISABLED)

self.images=[]

self.tk\_images=[]

self.imgrefs=[]

self.full\_imgs=[]

#self.textbox.grid(row=0,rowspan=100, column=1, padx=5, pady=5)

######################################################

#button = Button(self.textbox, text="Click", command=None)

#self.textbox.window\_create(INSERT, window=button)

######################################################

self.help="""Right click in the chat to costumize.

Left click on pictures to save or open them in default size.

Basic Controls

\* Arrow Left: Move left

\* Arrow Right: Move Right

\* Arrow Down: Soft Drop

\* Arrow Up: Rotate clockwise

\* Left Ctrl: Rotate counter-clockwise

\* Space: Hard Drop

\* C, Left shift: Hold

\* F11: Fullscreen mode \*(multiplayer only)\*

\* Escape: Exit Fullscreen mode \*(multiplayer only)\*"""

self.message=StringVar()

self.e\_out=Entry(self, width=maxchar-1, textvariable=self.message,font=self.font, bd=3,highlightbackground=self.foreground, highlightcolor=self.selectbackground, highlightthickness=2)

self.e\_out.grid(row=2, column=0,sticky=E)

self.e\_out.bind('<Return>', self.sendmsg)

self.w\_frame=Frame(self)

self.b\_settings=ttk.Button(self.w\_frame, text="Connection Settings", command=self.settings, state=NORMAL, width=27)

self.b\_disconnect=ttk.Button(self.w\_frame, text="Disonnect", command=self.disconnect, state=DISABLED)

self.b\_connect=ttk.Button(self.w\_frame, text="Connect", command=self.connect, state=NORMAL)

self.b\_out=ttk.Button(self, text="Send message", command=self.sendmsg, state=NORMAL)

self.b\_picout=ttk.Button(self, text="Send image", command=self.img\_send, state=NORMAL)

self.b\_help=ttk.Button(self.w\_frame, text="Help", command=lambda: messagebox.showinfo('Help', self.help), state=NORMAL)

self.b\_settings.grid(row=0, column=2, padx=10, pady=2, sticky=N+W)

self.b\_disconnect.grid(row=1, column=2, padx=10, pady=2, sticky=N+E)

self.b\_connect.grid(row=1, column=2, padx=10, pady=2, sticky =N+W)

self.b\_out.grid(row=2, column=2, padx=10, sticky =N+W)

self.b\_picout.grid(row=2, column=2, padx=10, sticky=N+E)

self.b\_help.grid(row=3, column=2, padx=10, sticky=N+W)

self.license=Label(self.w\_frame, text="©Áron L. Hertendi, 2018-2020")

self.license.grid(row=2, column=2, padx=10, sticky =N+W)

self.prog\_lab\_sv=StringVar(self)

self.prog\_iv=IntVar(self)

self.prog\_label=Label(self.w\_frame, textvariable = self.prog\_lab\_sv)

self.prog\_label.grid(row=3, column=2,sticky =N+E)

self.prog=ttk.Progressbar(self.w\_frame, orient="horizontal", length=200, mode="determinate", variable=self.prog\_iv)

self.prog.grid(row=5, column=2, pady=3,sticky =N)

self.w\_frame.grid(row=0,column=2,rowspan=1, pady=5,sticky =N)

#FONTDIRS = [os.path.join(os.environ['WINDIR'], 'Fonts')]

#for x in FONTDIRS:

#self.write((os.environ['WINDIR'], 'Fonts'))

self.grid(row=0, column=0,padx=5, pady=0)

self.connected = False

#self.connect()

def popup(self, event):

"""Rightclick on the chat log event handler"""

menu = Menu(self, tearoff=0)

menu.add\_command(label="Background color", command= lambda: self.set\_layout('b'))

menu.add\_command(label="Text color", command= lambda: self.set\_layout('f'))

menu.add\_command(label="Selected text background color", command= lambda: self.set\_layout('sb'))

menu.add\_command(label="Selected text color", command= lambda: self.set\_layout('sf'))

menu.add\_command(label="Font", command= lambda: self.set\_layout('font'))

menu.post(event.x\_root, event.y\_root)

def set\_layout(self, ground):

"""Reconfigure the textbox"""

if ground=='f':

self.foreground = tkinter.colorchooser.askcolor()[1]

elif ground=='b':

self.background= tkinter.colorchooser.askcolor()[1]

elif ground=='sb':

self.selectbackground= tkinter.colorchooser.askcolor()[1]

elif ground=='sf':

self.selectforeground= tkinter.colorchooser.askcolor()[1]

elif ground=='font':

SetFontTk(self,self.font)

self.textbox.config(foreground=self.foreground, background=self.background, selectbackground=self.selectbackground, selectforeground=self.selectforeground)

def set\_font(self, font):

"""Set the new font in the chat"""

self.font=font

self.textbox.config(font=self.font)

self.e\_out.config(font=self.font)

def disconnect(self):

"""Disconnect from the server."""

try:

self.send(bytes('#fin#', 'utf-8'))

self.write('Lekapcsolódott.')

except Exception as e:

pass

try:

self.master.drop\_connection()

except AttributeError:

pass

self.connected=False

self.b\_settings.config(state=NORMAL)

self.b\_connect.config(state=NORMAL)

self.b\_disconnect.config(state=DISABLED)

self.b\_out.config(state=DISABLED)

self.b\_picout.config(state=DISABLED)

def connect(self):

"""Connect to server"""

self.connection=socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.write("Connecting... - IPv4: {0}, Port: {1}".format(self.data[0].get(),self.data[1].get()))

HOST, PORT=self.data[0].get(),int(self.data[1].get())

try:

self.connection.connect((HOST, PORT))

except socket.error as e:

self.write(str(e))

self.write("Couldn't reach server.")

self.disconnect()

return

except Exception as e:

print(e)

self.connection.send(bytes(self.data[2].get(), 'utf-8'))

asd=self.connection.recv(1024).decode('utf-8')

if asd=='no':

self.write("Connection rejected. Server is already full.")

return

self.master.set\_connection(self.connection)

self.write("Successful connection.")

print(asd)

self.data[2].set(asd)

self.connected=True

self.e\_out.focus()

self.b\_settings.config(state=DISABLED)

self.b\_disconnect.config(state=NORMAL)

self.b\_out.config(state=NORMAL)

self.b\_picout.config(state=NORMAL)

self.b\_connect.config(state=DISABLED)

self.th\_rec=ThreadReception(self.connection, self)

#self.th\_emis=ThreadImageOut(self.connection, self)

self.th\_rec.start()

self.send(bytes('help.get', 'utf-8'))

#self.th\_emis.start()

def prog\_setter(self,x, length):

"""Set the progression of the progressbar, while receiving an image."""

self.prog\_iv.set(x)

self.prog\_lab\_sv.set("{0} %".format(int(x/length\*100))) #{:.2f}

if (x/length)==1:

self.prog\_lab\_sv.set(self.prog\_lab\_sv.get()+" processed!")

def prog\_getter(self):

"""Getter for the progressbar state"""

return self.prog\_iv.get()

def sendmsg(self, event=None):

"""Send a message to the chat"""

if self.connected:

if len(self.message.get())>0:

self.write(self.data[2].get()+'> '+self.message.get())

self.send(bytes(self.message.get(), 'utf-8'))

self.message.set('')

else:

self.write('Sending message unsuccessful, please connect...')

def img\_send(self):

"""Send an image to the chat"""

imgfile = filedialog.askopenfilename(initialdir = self.initdir,title = "Select file",filetypes = (("jpeg files","\*.jpg"),("png files","\*.png")))

if imgfile!='':

self.initdir=imgfile

mode, size=Image.open(imgfile).mode, Image.open(imgfile).size

self.images.append(Image.open(imgfile).tobytes())

self.sending=self.images[-1]

self.send(bytes("#pic#"+str(len(self.sending))+"#pic#"+str(mode)+"#pic#"+str(size[0])+"#pic#"+str(size[1]), 'utf-8'))

self.prog['maximum']=len(self.sending)

send\_th=ThreadImageOut(self.connection, self, self.sending)

send\_th.start()

#self.send(sending)

self.img\_show(imgfile, mode, size[0], size[1],self.data[2].get())

def img\_show(self, imgfile, mode, size0, size1, name):

"""Method to display the image in the chat"""

self.write('')

self.full\_imgs.append(Image.frombytes(mode,(size0,size1),self.images[-1]))

self.imgTk=ImageTk.PhotoImage(image=self.resized(self.full\_imgs[-1]))

self.tk\_images.append(self.imgTk)

##### INSERTION

self.imgrefs.append((self.textbox.image\_create(END, image=self.imgTk),len(self.imgrefs), imgfile))

self.textbox.tag\_add('pic'+str(self.imgrefs[-1][1]), self.imgrefs[-1][0])

self.textbox.config(state=NORMAL)

self.textbox.yview\_moveto(1.0)

self.textbox.insert(END, ' ('+name+')')

self.textbox.config(state=DISABLED)

x=self.imgrefs[-1]

self.textbox.tag\_bind('pic'+str(self.imgrefs[-1][1]), '<Button-1>', lambda func: self.img\_popup(func, x))

def img\_popup(self, event, img):

"""Method to handle Left-Click on images."""

menu = Menu(self, tearoff=0)

menu.add\_command(label="Show", command= lambda: self.img\_show\_full(img[1]))

menu.add\_command(label="Save", command= lambda: self.img\_save(img))

menu.post(event.x\_root, event.y\_root)

def img\_show\_full(self, img):

"""Method to open image in the OS's default image inspector"""

self.full\_imgs[img].show()

self.textbox.yview\_moveto(1.0)

self.textbox.mark\_set(INSERT, END)

def img\_save(self,img):

"""Method to save the image to the computer."""

filepath= filedialog.asksaveasfilename(parent=self,initialfile=img[2], defaultextension="\*.jpg", initialdir = self.initdir,title = "Choose dir",filetypes = (("jpeg files","\*.jpg"),("png files","\*.png"),("all files","\*.\*")))

if bool(filepath):

self.initdir=filepath

self.full\_imgs[img[1]].save(fp=filepath)

self.write("Mentve: "+filepath)

def resized(self, img):

"""Resize the image, to fit in the chatbox"""

if img.width>self.textbox.winfo\_width():

multip=img.width/self.textbox.winfo\_width()

size = int(img.width/multip), int(img.height/multip)

img=img.resize(size)

if img.height>self.textbox.winfo\_height():

multip=img.height/self.textbox.winfo\_height()

size = int(img.width/multip), int(img.height/multip)

img = img.resize(size)

return img

def write(self, text):

"""Any chatlogs should be inserted to the textbox by this method"""

self.textbox.config(state=NORMAL)

self.textbox.insert(END, '\n'+text)

self.textbox.yview\_moveto(1.0)

self.textbox.config(state=DISABLED)

def \_delete\_window(self):

"""On destroying the window"""

try:

self.send(bytes('#fin#', 'utf-8'))

except Exception as e:

pass

def settings(self):

"""Toplevel window to costumize connection settings"""

self.costumize=Toplevel(self)

self.costumize.grab\_set()

labels=['IP cím: ','Port: ', 'Név: ']

for x in range(3):

labels[x]=Label(self.costumize, text = labels[x])

labels[x].grid(row=x, column=0,sticky=E)

entries=[0]\*3

entry\_svar=[StringVar(self.costumize),StringVar(self.costumize),StringVar(self.costumize)]

for x in range(3):

entries[x]=Entry(self.costumize, textvariable=entry\_svar[x])

entry\_svar[x].set(self.data[x].get())

entries[x].grid(row=x, column=1)

b\_confirm=ttk.Button(self.costumize, command= lambda :self.save(self.costumize, entry\_svar), text="OK")

b\_confirm.grid(row=1, column=3, padx=10)

self.costumize.title('Settings')

self.costumize.bind("<Return>", lambda evt:self.save(self.costumize, entry\_svar))

self.costumize.geometry("+%d+%d" % (self.master.winfo\_rootx()+50,

self.master.winfo\_rooty()+50))

self.costumize.protocol("WM\_DELETE\_WINDOW", self.set\_destroyed)

self.costumize.resizable(0,0)

entries[0].focus\_set()

self.costumize.transient(self.costumize.master)

def set\_destroyed(self):

"""Settings window closed without save"""

self.costumize.destroy()

self.write('Settings have not been modified.')

def save(self,window, settings):

"""Settings window closed with save"""

window.destroy()

x= [i.get() for i in self.data]

self.data=settings

for i in range(3):

if self.data[i].get()!=x[i]:

self.write("Settings modified: "+x[i]+" -> "+self.data[i].get())

def send(self,msg):

"""Send message to the server accordingly to the uniquely desinged protocol"""

self.connection.send(bytes(chr(0),'utf-8'))

self.connection.send(msg)

self.connection.send(bytes(chr(0),'utf-8'))

class SetFontTk(Toplevel):

"""Window returning tk.font.Font()"""

def \_\_init\_\_(self, master, fonts):

Toplevel.\_\_init\_\_(self,master)

self.master = master

self.font=fonts

self.grab\_set()

labels=['Betűtítpus: ','Méret: ', 'Stílus: ', '']

for x in range(4):

labels[x]=Label(self, text = labels[x])

labels[x].grid(row=x, column=0, sticky=E)

self.combos=[0]\*4

self.combo\_opts=[font.families(), [9,10,11,12,14,16,18,20],['normal', 'bold'],['roman', 'italic']]

self.combo\_names=['family', 'size', 'weight', 'slant']

self.entry\_svar=[StringVar(self),StringVar(self),StringVar(self),StringVar(self)]

for x in range(4):

self.combos[x]=ttk.Combobox(self, textvariable=self.entry\_svar[x], values=self.combo\_opts[x])

self.combos[x].grid(row=x, column=1)

self.entry\_svar[x].set(self.font.cget(self.combo\_names[x]))

b\_confirm=ttk.Button(self, command= self.end, text="OK")

b\_confirm.grid(row=4, column=1, padx=10)

self.title('Betűtípus')

self.geometry("+%d+%d" % (self.master.winfo\_rootx()+50,

self.master.winfo\_rooty()+50))

#self.protocol("WM\_DELETE\_WINDOW", self.set\_destroyed)

self.resizable(0,0)

self.transient(self.master)

def end(self):

"""Upon saving font configuration"""

self.font=font.Font(family=self.combos[0].get(), size=self.combos[1].get(), weight=self.combos[2].get(), slant=self.combos[3].get())

self.master.set\_font(self.font)

self.destroy()

class ThreadReception(threading.Thread):

"""Thread object for receiving messages"""

def \_\_init\_\_(self, conn, root):

threading.Thread.\_\_init\_\_(self)

self.conn = conn #ref to socket

self.root = root

self.setDaemon(True)

def run(self):

"""Coninously check for incoming packets and decode them accordingly to the protocol"""

while True:

start=self.conn.recv(1).decode('UTF-8')

if start!=chr(0):

print(start)

if start=="":break

else:

inbox=''

while True:

try:

curr=self.conn.recv(1).decode('UTF-8')

if curr==chr(0):

break

inbox+=curr

except Exception as e:

print("Log: ", curr, "\nError: ", e)

if inbox.startswith("#pic#"):

self.root.b\_picout.config(state=DISABLED)

img\_data=inbox.split("#pic#")

self.root.prog['maximum']=int(img\_data[1])

img=b''

a=0

while a<int(img\_data[1]):

msgImg=self.conn.recv(4096)

img+=msgImg

a=len(img)

self.root.prog\_setter(a, int(img\_data[1]))

self.root.images.append(img)

self.root.img\_show('new', str(img\_data[2]), int(img\_data[3]), int(img\_data[4]), img\_data[5])

self.root.b\_picout.config(state=NORMAL)

time.sleep(2)

self.root.prog\_setter(0,1)

#########CREATE PROGRESS WINDOW FOR EACH CLIENT TO SHOW PROGRESS

#self.root.prval=int(inbox.split("#prog#")[1])

elif inbox.startswith("#PLAYER#"):

name=inbox.split("#")[-1]

self.root.master.add\_player(name)

elif inbox.startswith("#DELETE#"):

name=inbox.split("#")[-1]

self.root.master.remove\_player(name)

elif inbox.startswith("#LEVEL#"):

level=inbox.split("#")[-1]

self.root.write("Game difficulcity: LEVEL %s"%level)

self.root.master.set\_level(int(level))

elif inbox.startswith("#MAXUSER#"):

num=inbox.split("#")[-1]

self.root.write("Maximum users: %s"%num)

self.root.master.set\_scale(int(num))

elif inbox.startswith("#GAME#READY#"):

name=inbox.split("#")[-1]

self.root.write(name+" is ready to play!")

self.root.master.set\_ready(name)

elif inbox.startswith("#GAME#ANNOUNCE#"):

name=inbox.split("#")[-1]

bonus=inbox.split("#")[-2]

self.root.write(name+" had a "+bonus)

self.root.master.set\_ready(name)

elif inbox.startswith("#GAME#WON#"):

inbox=inbox.split("#GAME#")

x=inbox[1]

name=x.split("#")[-1]

self.root.master.set\_player(name, x.split("#")[0:2])

#ANNOUNCE

self.root.write(name+" has won this round!")

elif inbox.startswith("#GAME#"):

inbox=inbox.split("#GAME#")

x=inbox[1]

name=x.split("#")[-1]

self.root.master.set\_player(name, x.split("#")[0:2])

else:

self.root.write(inbox)

##### pil.image.frombytes

self.conn.close()

# Received a final message

class ThreadImageOut(threading.Thread):

"""Class that handles image sending"""

def \_\_init\_\_(self, conn, root,msg):

threading.Thread.\_\_init\_\_(self)

self.conn = conn #ref to socket

self.root=root

self.msg=msg

self.msglen=len(msg)

def run(self):

a=0

self.root.e\_out.config(state=DISABLED)

self.root.b\_out.config(state=DISABLED)

self.root.b\_picout.config(state=DISABLED)

self.root.b\_disconnect.config(state=DISABLED)

while True:

self.root.prog\_setter(self.root.prog\_getter()+self.conn.send(self.msg[a\*4096:((a+1)\*4096)]), self.msglen)

if self.root.prog\_getter()==len(self.msg) :break

a+=1

self.root.b\_picout.config(state=NORMAL)

self.root.b\_out.config(state=NORMAL)

self.root.e\_out.config(state=NORMAL)

self.root.b\_disconnect.config(state=NORMAL)

time.sleep(2)

self.root.prog\_setter(0,1)

def chat\_main():

root=ChatGui(None,'erin-PC', '64164', socket.gethostname()+'\\'+getpass.getuser())

#root=ChatGui('aronsv.ddns.net', '45000', socket.gethostname()+'\\'+getpass.getuser())

root.mainloop()

####WRITEFILE

if \_\_name\_\_ == '\_\_main\_\_':

chat\_main()